МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Классификация бинарных отношений и системы замыканий**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студента 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Токарева Никиты Сергеевича

|  |  |  |
| --- | --- | --- |
| Проверил аспирант | \_\_\_\_\_\_\_\_\_\_ | В. Н. Кутин |
|  | подпись, дата |  |

Саратов 2022

**1 Постановка задачи**

Цель работы ­­– изучение основных свойств бинарных отношений и операций замыкания бинарных отношений.

Порядок выполнения работы:

1. Разобрать основные определения видов бинарных отношений и разработать алгоритмы классификации бинарных отношений.
2. Изучить свойства бинарных отношений и рассмотреть основные системы замыкания на множестве бинарных отношений.
3. Разработать алгоритмы построения основных замыканий бинарных отношений.

**2 Теоретические сведения по рассмотренным темам с их обоснованием**

**2.1 Определение видов бинарных отношений**

Подмножества декартова произведения множеств и называются **бинарными отношениями** между элементами множеств , и обозначаются строчными греческими буквами: , …,…

Для бинарного отношения область определения и множество значений определяется как подмножества соответствующих множеств и по следующим формулам:

для некоторого

для некоторого .

**2.2 Свойства бинарных отношений**

Бинарное отношение называется:

* рефлексивным, если ;
* антирефлексивным, если ;
* симметричным, если ;
* антисимметричным, если и ;
* транзитивным, если и ;
* антитранзитивным, если и .

Стоит отметить, что бинарные отношения можно классифицировать по вышеперечисленным свойствам:

* Бинарное отношение на множестве называют отношением квазипорядка, если оно рефлексивно и транзитивно;
* Бинарное отношение на множестве называют отношением эквивалентности (или просто эквивалентностью), если оно рефлексивно, симметрично и транзитивно;
* Бинарное отношение на множестве называют отношением частичного порядка, если оно рефлексивно, антисимметрично и транзитивно;
* Бинарное отношение на множестве называют отношением строгого порядка, если оно антирефлексивно, антисимметрично и транзитивно.

Пусть задано бинарное отношение и определена матрица данного бинарного отношения , тогда бинарное отношение называется

1. рефлексивным тогда и только тогда, когда . Это означает, что бинарное отношение рефлексивно, если , где - единичная матрица. Если же матрица несравнима с единичной матрицей, то бинарное отношение не является рефлексивным;
2. симметричным тогда и только тогда, когда . Это означает, что бинарное отношение симметрично, если , где – транспонированная матрица бинарного отношения . Если же матрица несравнима с , то бинарное отношение не является симметричным;
3. антисимметричным тогда и только тогда, когда . Это означает, что бинарное отношение антисимметрично, если (значения элементов вне главной диагонали матрицы равны нулю), где – операция поэлементного произведения матриц;
4. транзитивным тогда и только тогда, когда . Это означает, что бинарное отношение транзитивно, если .

**2.3 Системы замыканий**

Множество подмножеств множества называется **системой замыканий**, если оно замкнуто относительно пересечений, т.е. выполняется

для любого подмножества .

Лемма 1 (о системах замыканий бинарных отношений). На множестве всех бинарных отношений между элементами множества следующие множества являются системами замыканий:

1. – множество всех рефлексивных бинарных отношений между элементами множества ,
2. – множество всех симметричных бинарных отношений между элементами множества ,
3. – множество всех транзитивных бинарных отношений между элементами множества ,
4. – множество всех отношений эквивалентности на множестве .

Множествовсех антисимметричных бинарных отношений между элементами множества не является системой замыкания.

**Оператором замыкания** на множестве называется отображение множества всех подмножеств в себя, удовлетворяющее условиям:

1. ;
2. ;
3. ,

для всех ,. Для подмножества значение называется **замыканием** подмножества .

Лемма 2 (о замыканиях бинарных отношений). На множестве всех бинарных отношений между элементами множества следующие отображения являются операторами замыканий:

1. – наименьшее рефлексивное бинарное отношение, содержащее отношение ;
2. – наименьшее симметричное бинарное отношение, содержащее отношение ;
3. – наименьшее транзитивное бинарное отношение, содержащее отношение ;
4. – наименьшее отношение эквивалентности, содержащее отношение .

**3 Результаты работы**

* 1. **Описание алгоритмов классификаций бинарных отношений**

Алгоритм 1 – Алгоритм проверки бинарного отношения на рефлексивность

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Булева переменная, принимающая значение «true», если бинарное отношение обладает свойством рефлексивности, и «false», если не обладает этим свойством.

Шаг 1. Инициализировать булеву переменную: .

Шаг 2. Необходимо проверить все элементы матрицы бинарного отношения, находящиеся на главной диагонали, т.е. элементы , где . Если , то присвоить и перейти к шагу 3, иначе перейти к проверке следующего элемента , где .

Шаг 3. Вернуть значение переменной в качестве выхода функции.

Оценка сложности алгоритма проверки бинарного отношения на рефлексивность определяется как .

Алгоритм 2 – Алгоритм проверки бинарного отношения на антиреф-лексивность

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Булева переменная, принимающая значение «true», если бинарное отношение обладает свойством антирефлексивности, и «false», если не обладает этим свойством.

Шаг 1. Инициализировать булеву переменную: .

Шаг 2. Необходимо проверить все элементы матрицы бинарного отношения, находящиеся на главной диагонали, т.е. элементы , где . Если , то присвоить и перейти к шагу 3, иначе перейти к проверке следующего элемента , где .

Шаг 3. Вернуть значение переменной в качестве выхода функции.

Оценка сложности алгоритма проверки бинарного отношения на антирефлексивность определяется как .

Алгоритм 3 – Алгоритм проверки бинарного отношения на симметричность

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Булева переменная, принимающая значение «true», если бинарное отношение обладает свойством симметричности, и «false», если не обладает этим свойством.

Шаг 1. Инициализировать булеву переменную .

Шаг 2. Построить матрицу путем транспонирования матрицы .

Шаг 3. Если , где – элементы матрицы , – элементы матрицы ,, то присвоить и перейти к шагу 4, иначе перейти к проверке следующих элементов и () матриц и соответственно.

Шаг 4. Вернуть значение переменной в качестве выхода функции.

Оценка сложности алгоритма проверки бинарного отношения на симметричность будет состоять из оценки сложности выполнения транспонирования матрицы бинарного отношения, которая равна , и оценки сложности прохождения по всем элементам и () матриц и соответственно равная . Тогда оценка сложности данного алгоритма равна .

Алгоритм 4 – Алгоритм проверки бинарного отношения на антисим-метричность

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Булева переменная, принимающая значение «true», если бинарное отношение обладает свойством антисимметричности, и «false», если не обладает этим свойством.

Шаг 1. Инициализировать булеву переменную .

Шаг 2. Построить матрицу путем транспонирования матрицы

Шаг 3. Если , где – элементы матрицы , – элементы матрицы ,, то присвоить и перейти к шагу 4, иначе перейти к проверке следующих элементов и () матриц и соответственно.

Шаг 4. Вернуть значение переменной в качестве выхода функции.

Оценка сложности алгоритма проверки бинарного отношения на антисимметричность будет состоять из оценки сложности выполнения транспонирования матрицы бинарного отношения, которая равна , и оценки сложности прохождения по всем элементам и ( матриц и соответственно равная . Тогда оценка сложности данного алгоритма равна .

Алгоритм 5 – Алгоритм проверки бинарного отношения на транзитивность

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Булева переменная, принимающая значение «true», если бинарное отношение обладает свойством транзитивности, и «false», если не обладает этим свойством.

Шаг 1. Инициализировать булеву переменную .

Шаг 2. Построить матрицу путем возведения матрицы бинарного отношения в квадрат (т.е. .

Шаг 3. Сравнить каждый элемент матрицы с каждым элементом матрицы . Если , где , то присвоить и перейти к шагу 4, иначе перейти к проверке следующих элементов и ( матриц и соответственно.

Шаг 4. Вернуть значение переменной в качестве выхода функции.

Оценка сложности алгоритма проверки бинарного отношения на транзитивность будет состоять из оценки сложности выполнения возведения матрицы в квадрат, которая равна , и оценки сложности прохождения по всем элементам и ( матриц и соответственно равная . Тогда оценка сложности данного алгоритма равна .

Алгоритм 6 – Алгоритм вывода свойств и классификации бинарного отношения

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* «Бинарное отношение является рефлексивным», «Бинарное отношение является антирефлексивным», «Бинарное отношение не является рефлексивным и антирефлексивным», «Бинарное отношение является симметричным», «Бинарное отношение является антисимметричным», «Бинарное отношение не является симметричным и антисимметричным», «Бинарное отношение является транзитивным», «Бинарное отношение не является транзитивным», «Бинарное отношение является отношением квазипорядка», «Бинарное отношение является отношением эквивалентности», «Бинарное отношение является отношением частичного порядка» или «Бинарное отношение является отношением строгого порядка».

Шаг 1. Инициализировать булевы переменные , , , , и присвоить им значения, получающихся при запуске алгоритмов 1, 2, 3, 4, 5, соответственно.

Шаг 2. Если , то вернуть значение «Бинарное отношение является рефлексивным», если , то вернуть значение «Бинарное отношение является антирефлексивным», и если , то вернуть значение «Бинарное отношение не является рефлексивным и антирефлексивным».

Шаг 3. Если , то вернуть значение «Бинарное отношение является симметричным», если , то вернуть значение «Бинарное отношение является антисимметричным», и если , то вернуть значение «Бинарное отношение не является симметричным и антисимметричным».

Шаг 4. Если , то вернуть значение «Бинарное отношение является транзитивным», если , то вернуть значение «Бинарное отношение не является транзитивным».

Шаг 5. Если , то вернуть значение «Бинарное отношение является отношением квазипорядка».

Шаг 6. Если , то вернуть значение «Бинарное отношение является отношением эквивалентности».

Шаг 7. Если , то вернуть значение «Бинарное отношение является отношением частичного порядка».

Шаг 8. Если , то вернуть значение «Бинарное отношение является отношением строгого порядка».

Оценка сложности с учётом вызова функций, в которых проверяются свойства бинарного отношения, равна .

* 1. **Описание алгоритмов построения замыканий**

Алгоритм 7 – Алгоритм построения замыкания рефлексивности

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Исходная матрица бинарного отношения, замкнутая относительно рефлексивности.

Шаг 1. Необходимо пройти по всем элементам матрицы , где , и присвоить им единицу (т.е.).

Оценка сложности данного алгоритма равна .

Алгоритм 8 – Алгоритм построения замыкания симметричности

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Исходная матрица бинарного отношения, замкнутая относительно симметричности.

Шаг 1. Необходимо пройти по всем элементам матрицы , где Если элемент , а элемент , то нужно присвоить единицу.

Оценка сложности данного алгоритма равна .

Алгоритм 9 – Алгоритм построения замыкания транзитивности

*Вход.* Квадратная матрица бинарного отношения размерности .

*Выход.* Исходная матрица бинарного отношения, замкнутая относительно транзитивности.

Шаг 1. Если , то нужно приравнять к единице, где , , – элементы матрицы , . Данная процедура должна повторится раз согласно пункту 3 леммы 2 (о замыканиях бинарных отношений), где говорится об операторе транзитивного замыкания.

Оценка сложности данного алгоритма равна .

* 1. **Код программы, реализующий рассмотренные алгоритмы**

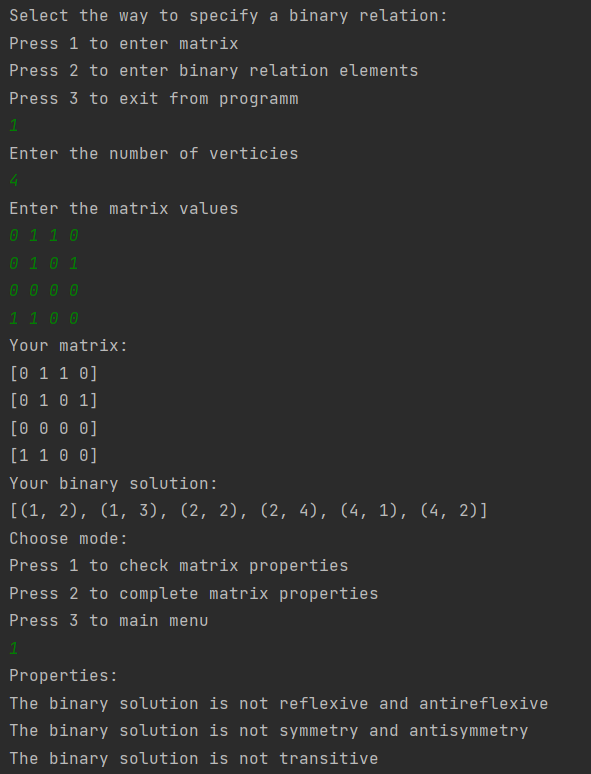
Далее показан код программы.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237 | **import** **numpy** **as** **np**  *# Проверка свойства рефлексивности*  **def** check\_reflexivity(a, n):  isReflexive = True  **for** i **in** range(n):  **if** a[i][i] == 0:  isReflexive = False  **return** isReflexive  **return** isReflexive  *# Проверка свойства антирефлексивности*  **def** check\_antireflexivity(a, n):  isAntireflexive = True  **for** i **in** range(n):  **if** a[i][i] != 0:  isAntireflexive = False  **return** isAntireflexive  **return** isAntireflexive  *# Проверка свойства симметричности*  **def** check\_symmetry(a, n):  isSymmetry = True  t = a.transpose()  **for** i **in** range(n):  **for** j **in** range(n):  **if** a[i][j] != t[i][j]:  isSymmetry = False  **return** isSymmetry  **return** isSymmetry  *# Проверка свойства антисимметричности*  **def** check\_antisymmetry(a, n):  isAntisymmetry = True  t = a.transpose()  **for** i **in** range(n):  **for** j **in** range(n):  **if** i != j **and** a[i][j] \* t[i][j] != 0:  isAntisymmetry = False  **return** isAntisymmetry  **return** isAntisymmetry  *# Проверка свойства транзитивности*  **def** check\_transitivity(a, n):  isTransitive = True  t = np.dot(a, a)  **for** i **in** range(n):  **for** j **in** range(n):  **if** t[i][j] > 1:  t[i][j] = 1  **if** a[i][j] < t[i][j]:  isTransitive = False  **return** isTransitive  **return** isTransitive  *# Добавление свойства рефлексивности*  **def** insert\_reflexivity(a, n):  **for** i **in** range(n):  a[i][i] = 1  **return** construction\_of\_closures(a, n)  *# Добавление свойства симметричности*  **def** insert\_symmetry(a, n):  **for** i **in** range(n):  **for** j **in** range(n):  **if** a[i][j] == 1 **and** a[i][j] != a[j][i]:  a[j][i] = 1  **return** construction\_of\_closures(a, n)  *# Добавление свойства транзитивности*  **def** insert\_transitivity(a, n):  **for** m **in** range(n):  **for** i **in** range(n):  **for** j **in** range(n):  **for** k **in** range(n):  **if** a[i][k] **and** a[k][j]:  a[i][j] = 1  **return** construction\_of\_closures(a, n)  *# Проверка матрицы на свойства и их классификация*  **def** testing\_of\_properties(a, n):  bl\_ref = check\_reflexivity(a, n)  bl\_aref = check\_antireflexivity(a, n)  bl\_sym = check\_symmetry(a, n)  bl\_asym = check\_antisymmetry(a, n)  bl\_tran = check\_transitivity(a, n)  **print**('Properties:')  **if** bl\_ref:  **print**('The binary solution is reflexive')  **else**:  **if** bl\_aref:  **print**('The binary solution is antireflexive')  **else**:  **print**('The binary solution is not reflexive and antireflexive')  **if** bl\_sym:  **print**('The binary solution is symmetry')  **else**:  **if** bl\_asym:  **print**('The binary solution is antisymmetry')  **else**:  **print**('The binary solution is not symmetry and antisymmetry')  **if** bl\_tran:  **print**('The binary solution is transitive')  **else**:  **print**('The binary solution is not transitive')  **if** bl\_ref **and** bl\_tran:  **print**('The binary relation is quasi-order')  **if** bl\_ref **and** bl\_sym **and** bl\_tran:  **print**('The binary relation is equivalence')  **elif** bl\_ref **and** bl\_asym **and** bl\_tran:  **print**('The binary relation is partial order')  **elif** bl\_aref **and** bl\_asym **and** bl\_tran:  **print**('The binary relation is strict order')  **return** choose\_mode(a, n)  *# Построение замыканий*  **def** construction\_of\_closures(a, n):  print\_matrix(a, n)  print\_binary\_solution(a, n)  **print**('Choose one of the following expression')  **print**('Press 1 to complete the reflexivity property')  **print**('Press 2 to complete the symmetry property')  **print**('Press 3 to complete the transitivity property')  **print**('Press 4 to exit')  bl = int(input())  **if** bl == 1:  insert\_reflexivity(a, n)  **elif** bl == 2:  insert\_symmetry(a, n)  **elif** bl == 3:  insert\_transitivity(a, n)  **elif** bl == 4:  choose\_mode(a, n)  **else**:  **print**('Incorrect input')  construction\_of\_closures(a, n)  *# Выбор способа задачи бинарного отношения*  **def** choose\_mode(a, n):  **print**('Choose mode:')  **print**('Press 1 to check matrix properties')  **print**('Press 2 to complete matrix properties')  **print**('Press 3 to main menu')  bl = int(input())  **if** bl == 1:  testing\_of\_properties(a, n)  **elif** bl == 2:  construction\_of\_closures(a, n)  **elif** bl == 3:  **return** launch()  **else**:  **print**('Incorrect output')  *# Вывод матрицы*  **def** print\_matrix(a, n):  cnt = 0  **print**('Your matrix:')  **for** i **in** a:  **if** (cnt < n):  **print**(f'{i}' + ' ')  **else**:  **print**(f'{i}' + '**\n**')  cnt = 0  cnt += 1  *# Вывод бинарного отношения*  **def** print\_binary\_solution(a, n):  br = []  **print**('Your binary solution:')  **for** i **in** range(n):  **for** j **in** range(n):  **if** a[i][j] == 1:  br.append((i + 1, j + 1))  **print**(br)  *# Главное меню*  **def** launch():  **print**('Select the way to specify a binary relation:')  **print**('Press 1 to enter matrix')  **print**('Press 2 to enter binary relation elements')  **print**('Press 3 to exit from programm')  bl = int(input())  **if** bl == 1:  **print**('Enter the number of verticies')  n = int(input())  **print**('Enter the matrix values')  a = []  **for** i **in** range(n):  a.append([int(j) **for** j **in** input().split()])  a = np.array(a)  print\_matrix(a, n)  print\_binary\_solution(a, n)  choose\_mode(a, n)  **elif** bl == 2:  **print**('Enter an even amount of numbers on one line')  s = input()  **print**(s)  br = [int(i) **for** i **in** s.split(' ')]  **if** len(br) % 2 != 0:  **print**('Incorrect input')  launch()  **else**:  cnt = 0  n = max(br)  **print**(n)  a = []  **for** i **in** range(n):  a.append([0 **for** j **in** range(n)])  **for** i **in** range(len(br) // 2):  a[br[cnt] - 1][br[cnt + 1] - 1] = 1  cnt += 2  a = np.array(a)  print\_binary\_solution(a, n)  print\_matrix(a, n)  choose\_mode(a, n)  **elif** bl == 3:  **return**  **return**  launch() |

* 1. **Результаты тестирования программы**

Пример 1 – Ввод матрицы

На рисунке 1 представлен ввод матрицы размерности . Из рисунка видно, что прежде чем что-то ввести программа предлагает нам выбор способа задания бинарного отношения. Сначала воспользуемся 1 режимом, чтобы ввести матрицу. После ввода будет предложен выбор, где, нажав цифру 1, на экране отобразятся все свойства бинарного отношения, а также вид классификации, к которой оно относится.

 Рисунок 1 – Ввод матрицы и просмотр свойств бинарного отношения

На рисунке 2 показано построение замыкания текущей матрицы. В качестве эксперимента было решено достроить её относительно рефлексивности и транзитивности, нажав в консоли 2 и 3 соответственно, как показано на рисунке 3.

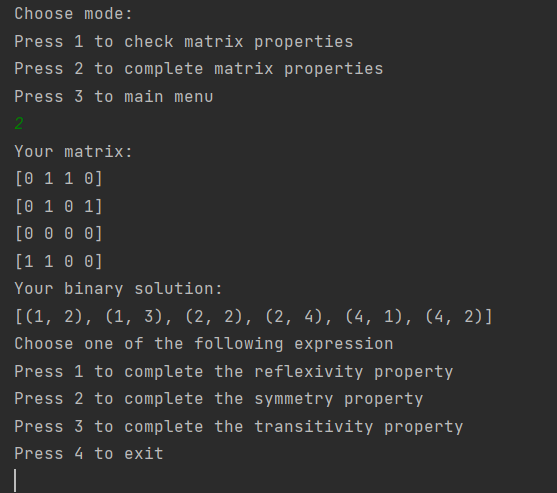
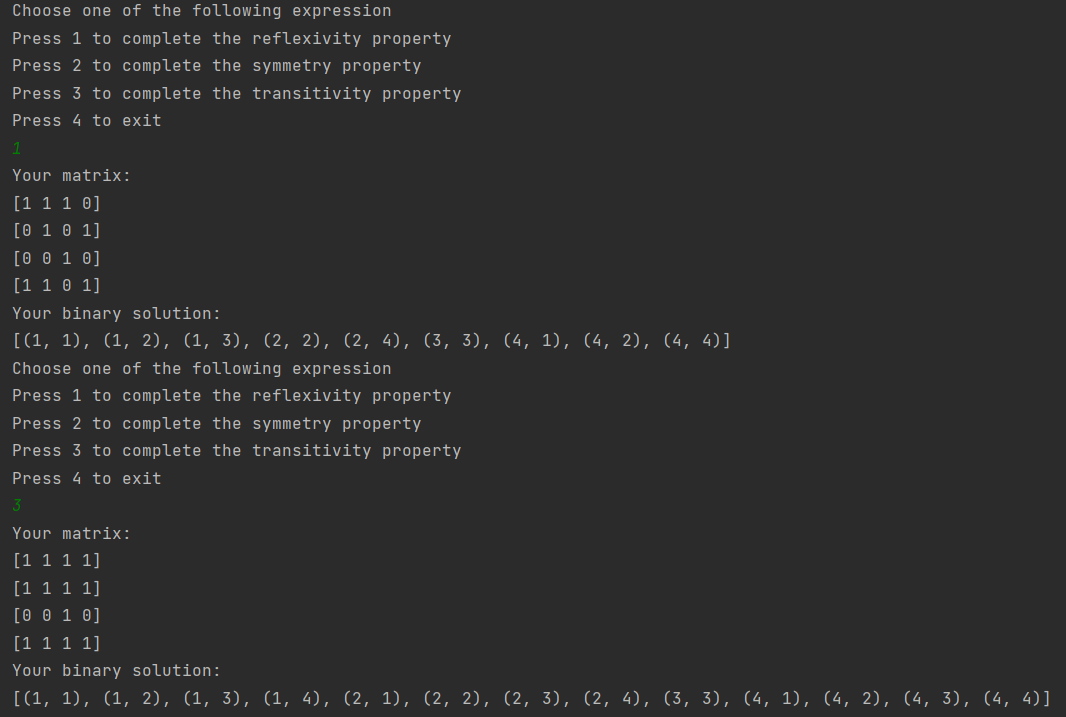
 Рисунок 2 – Выбор режима построения замыкания

Рисунок 3 – Построения замыкания относительно рефлексивности и транзитивности

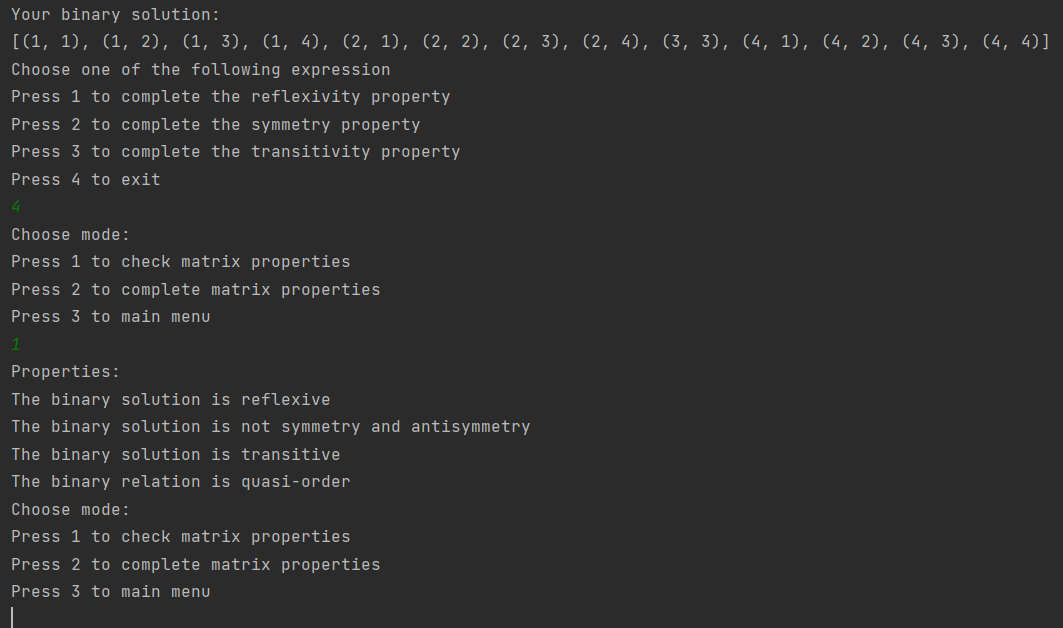
На рисунке 4 показан выхода из режима построения, а также вход в режим просмотра свойств матрицы бинарного отношения. Как видно из рисунка 4, преобразования были выполнены корректно.

Рисунок 4 – вывод свойств матрицы после преобразования

Пример 2 – Ввод бинарного отношения

На рисунке 5 показан ввод бинарного отношения, где необходимо ввести все числа через пробел.

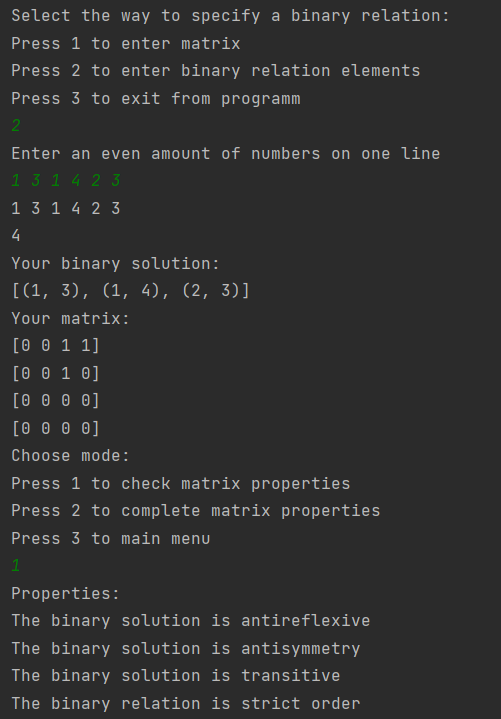
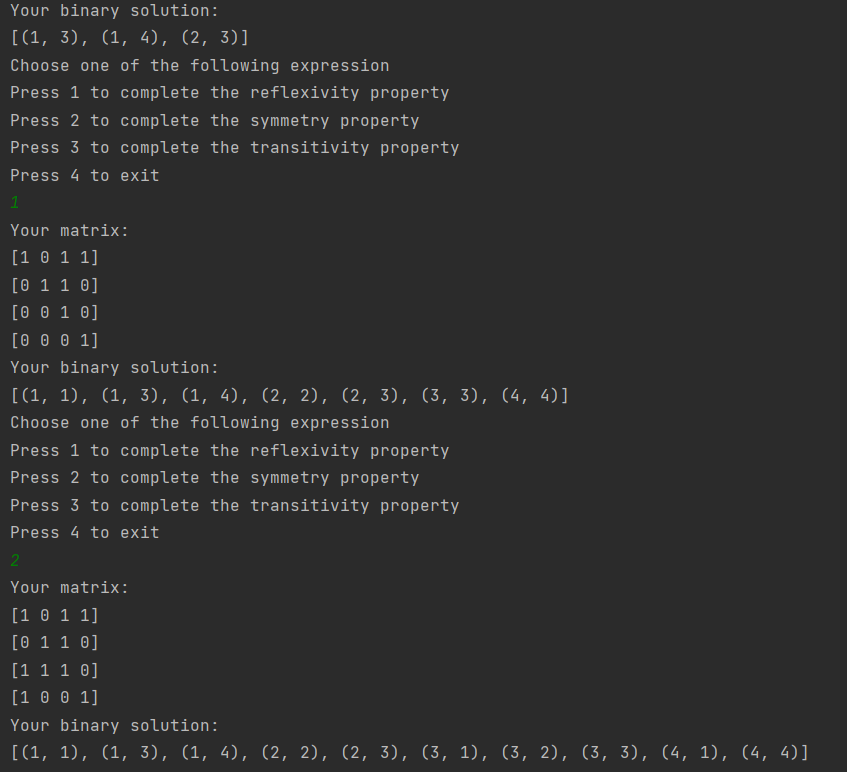


Рисунок 5 – ввод бинарного отношения

На рисунках 6 и 7 показано построение матрицы до эквивалентности.

 Рисунок 6 – Построения замыкания относительно рефлексивности и симметричности

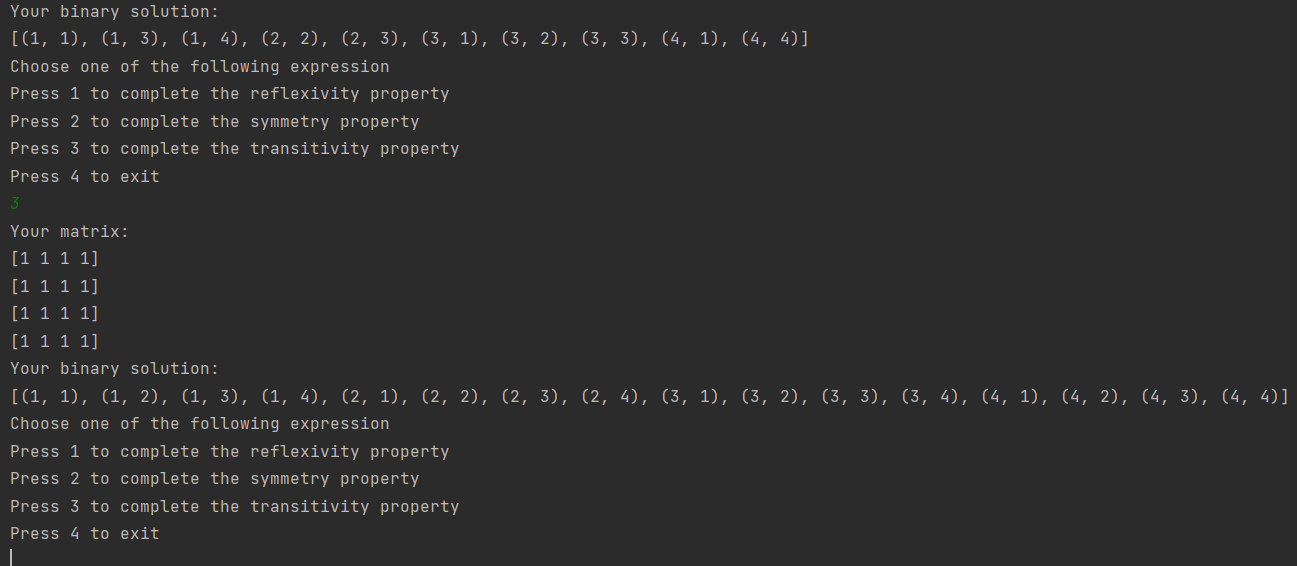
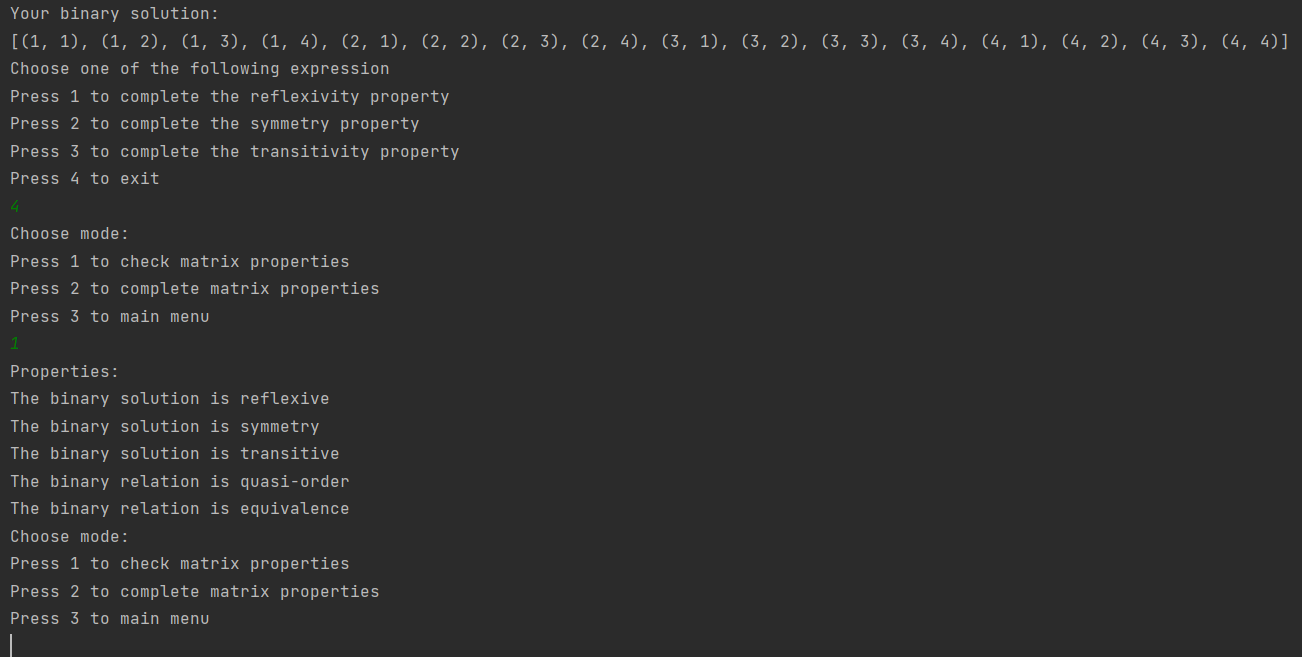


Рисунок 7 – Построения замыкания относительно транзитивности

На рисунке 8 показана проверка свойств бинарного отношения. Как видно из данного рисунка, всё работает корректно.

 Рисунок 8 – Просмотр свойств после преобразований текущего бинарного отношения

**ЗАКЛЮЧЕНИЕ**

В результате лабораторной работы были рассмотрены теоретические сведения о свойствах бинарных отношений, их виды построения замыканий относительно рефлексивности, симметричности, транзитивности. Опираясь на изложенную выше теорию, были разработаны алгоритмы классификации и определения свойств бинарных отношений, а также алгоритмы построения замыканий по тому или иному свойству. Была произведена оценка сложности каждого из построенных алгоритмов. Была реализована программа, написанная на языке Python, с использованием библиотеки NumPy для упрощения работы с матрицами и массивами.